APPENDIX

A. Comparing associational and causal attributions: single-concept toy example

Setup Consider a simple example where the signal is represented by a sine wave with amplitude $a \in [0, 2]$. We assume that $a \geq 1$ corresponds to the positive and $a < 1$ to the negative class. More specifically, we assume that the classifiers output probability is given by $f(a) = \frac{a}{2}$. Furthermore, the data generative process is parametrized via

$$p(a) = \begin{cases} p & \text{if } a \in [0, 1] \\ 1 - p & \text{if } a \in [1, 2] \,. \end{cases}$$

Here, the parameter p allows to induce a class imbalance between the positive and the negative class.

Causal Attribution We can now work out the causal attribution as follows

$$\begin{aligned} ITE &= \int_1^2 \log_2(\frac{a}{2})\mathrm{d}a - \int_1^2 \log_2(\frac{a}{2})\mathrm{d}a \\ &= 1 - \frac{1}{\log 2} + 1 + \frac{1}{\log 2} = 2 \,. \end{aligned} \tag{6}$$

It turns out to be positive irrespective of the value a in the input sample under consideration.

Associational Attribution Similarly, we can work out the associational attribution for a sample with parameter value a:

$$\begin{aligned} IAA &= \log_2 f(a) - \int_0^2 p(a) f(a) \mathrm{d}a \\ &= \log_2 \frac{a}{2} - p\left(-1 - \frac{1}{\log 2}\right) - (1 - p)\left(1 - \frac{1}{\log 2}\right) . \end{aligned} \tag{7}$$

Let us now consider the attribution in the case of a sample from the positive class, i.e., $a \geq 1$. The first terms is minimal for $a = 1$ and it is straightforward to work out that the attribution changes sign within the positive class if $p > 1 - \frac{1}{2 \log 2} \approx 0.279$, i.e., in the case where the data distribution shows a sufficient imbalance towards the positive class.

Conclusion We argue that this sign change in the association attribution is an unnatural behaviour for the classifier defined above, which is not observed in the case of the causal attribution. We fully acknowledge that this is a very simple example and that more complicated examples should involve correlations between features/concepts and consider imperfect classifiers that at least partially leverage such correlations.

B. Computational complexity

TABLE I: Computational complexity

Description	Train[h]	Generate[s]
Drought	18.5	21
PTB-XL	18.9	21
Schizophrenia	19.2	21

Tab. I contains the details of the computational complexity of the proposed approach. We present the training in hours and sampling in seconds of each dataset separately as certain attributes differ from each other, such as the sample length, and the number of channels of each time series, similarly, one has to consider the computational power in use, in this case, the model training was executed on separate NVIDIA L40 GPUs, each with 48GB VRAM and around 18,176 CUDA cores, supported by 16GB RAM and 16-CPU cores. The results of the generation column represent the time for the imputation of a single concept for the class.

C. Datasets

TABLE II: Datasets details

Description	Drought	PTB-XL	Schizophrenia
Train size	300,000	9,754	1,980
Validation size	165,638	1,226	270
Test size	169,450	1,232	270
Classifier batch	32	32	32
Imputer batch	6	6	6
Sample length	180	248	248
Sample features	18	12	16
Classes	2	2	2
Concepts	4	6	4

Tab. II provides details for the three considered datasets: Drought, PTB-XL, and Schizophrenia. It includes information on the size of the training, validation, and test sets, batch sizes for both classifier and imputer models, sample length, number of sample features, classes, and concepts present in each dataset. To avoid data leakage, the drought dataset is split by the provided time horizons from past to present into train, val, and test, whereas the PTB-XL and schizophrenia datasets are split patient-wise. For the Drought dataset, concepts were generated using k-means with elbow-method leveraging the implementation from sci-kit learn. PTB-XL utilized well-defined concepts from existing literature [19]. Meanwhile, for the schizophrenia dataset, we employed a micro-states open-source software that is internally based on k-means, where based on the elbow method we select 4 microstate concepts.

D. Models

TABLE III: S4 hyperparameters

Hyperparameter	Value
Block of layers	4
s4 model copies	512
s4 state size	8
Optimizer	Adam
Learning rate	0.001
Weight decay	0.001
learning rate schedule	constant
Batch size	64
Epochs	20

Tab. III outlines the hyperparameters employed in the S4 model. The architecture consists of four blocks of layers, with each block containing 512 copies of the S4 model. The state size within the S4 model is set to 8. For optimization, the Adam optimizer is utilized with a learning rate and weight decay

both set to 0.001. The learning rate schedule is maintained constant throughout training. A batch size of 64 samples is used for each training iteration, spanning a total of 20 epochs. The training objective is to minimize the binary cross-entropy loss. During training, we apply a model selection strategy on the best performance (AUROC) on the validation set.

TABLE IV: Diffusion model hyperparameters

Hyperparameter	**Value**
Residual layers	36
Residual channels	256
Skip channels	256
Diffusion embedding dim. 1	128
Diffusion embedding dim. 2	512
Diffusion embedding dim. 3	512
Schedule	Linear
Diffusion steps T	200
B_0	0.0001
B_1	0.02
Optimizer	Adam
Loss function	MSE
Learning rate	0.0002
S4 state N dimensions	64
S4 bidirectional	Yes
S4 layer normalization	Yes
S4 Drop-out	0.0
S4 Maximum length	as required

Tab. IV present the hyperparameters and training approach for the CausalConceptTS model. Built upon *DiffWave* [43], and previously presented as *SSSD* [33] our model consists of 36 residual layers with 256 channels. It integrates a three-layer diffusion embedding (128, 256, and 256 dimensions) with swish activations, followed by convolutional layers. Our diffusion spans 200 time steps, using a linear schedule from 0.0001 to 0.02 for beta. We optimize with Adam (LR: 0.0002). Based on previous works [33] we trained each model over 50,000 iterations with model selection on lower MSE loss every 1,000 iterations. For the S4 model, we utilize a bidirectional layer with layer normalization, no dropout, and internal state dimension $N = 64$. This S4 layer captures bidirectional time dependencies. We maintain layer normalization and an internal state of $N = 64$, consistent with prior work [28].

In this work, we trained class-specific imputer models, one for each condition under consideration. An obvious alternative might seem to be to train a class-conditional imputer model. However, this requires to specify a procedure to adjust the importance of the class-conditional input within the framework of classifier-free guidance. While we observed minimal effects on dropout rates during training, increasing the alpha parameter during sampling improved imputation and causal effects but resulted in unrealistic time series, like excessively large R peaks in ECG data. As a result, we decided to base our experiments on class-specific imputation models.